DOCUMENT RESUME

ED 060 625 EM 009 631

AUTHOR	Bork, Alfred M.
TITLE	Inexpensive Timeshared Graphics on the SIGMA 7.
INSTITUTION	California Univ., Irvine. Physics Computer Development Project.
SPONS AGENCY	National Science Foundation, Washington, D.C.
PUB DATE	29 Sep 71
NOTE	21p.; Paper presented at the XDS Users Group International Meeting (17th, Las Vegas, Nevada, September 29, 1971)
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	*Computer Assisted Instruction; *Computer Graphics; Computer Programs; *Display Systems; Physics Instruction; Programed Instruction; Programing; Programing Languages; Time Sharing
IDENTIFIERS	Adage 100; FORTRAN; Sigma 7; Tektronix 4002 4010

ABSTRACT

This paper gives a technical description of various computer graphics programs developed on the Sigma 7 computer. Terminals used are the Adage 100 and the Tektronix 4002-4010. Commands are Metasymbol procedures which access Metasymbol library subroutines; programs can also be coupled with FORTRAN programs. Available, inexpensive graphic terminals are reviewed. Relatively minor changes are required to adapt the coding to terminals using different graphic coding. Graphics software is discussed, and the applications of graphics to teaching physics is emphasized. (RB)

ED 060625

INEXPENSIVE TIMESHARED GRAPHICS ON THE SIGMA 7

XDS Users Group
17th International Meeting - Las Vegas, Nevada

Alfred M. Bork
Physics Computer Development Project
University of California, Irvine

September 28, 1971

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

Abstract

This paper describes graphics programs using the Adage ARDS 100 or Tektronix 4002-4010 Terminals, developed for BTM and UTS on the Sigma 7. As with previous software development on this project, graphic commands are Metasymbol procedures which access Metasymbol library subroutines. Programs can be coupled with FORTRAN programs. Examples of the commands available and of graphic usage in teaching environments are discussed. Relatively minor changes are required to adapt the coding to terminals using different graphic coding.

I propose to tell you about our recent use of inexpensive graphic terminals in a timeshared environment on the Sigma 7, both in BTM and UTS. Inexpensive terminals I take to mean $10,000 or less. The environment is not dedicated; most of the terminals

EM007631

on the machi
project has
based physic
but to be ab
extensive gr

First, I wil
Then I will
sary to supp
software, an
in teaching

Review of I
The last fe
of graphics
is likely t
do any type
over $100,0
costs under
of a fancy
very usable
numeric abi

Perhaps the
based on th
storage is

THE SIGMA 7

egas, Nevada

t

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

rams using the Adage ARDS
ls, developed for BTM and
ous software development on
Metasymbol procedures which
nes. Programs can be coupled
f the commands available and
onments are discussed. Relative-
dapt the coding to terminals

ecent use of inexpensive graphic
ment on the Sigma 7, both in
is I take to mean $10,000 or
icated; most of the terminals

on the machine are alphanumeric, either CRT's or hardcopy. Our project has been concerned with the production of computer-based physics teaching materials, as you will see from the examples, but to be able to use a computer in this way we had to develop extensive graphic capabilities.

First, I will briefly describe available graphic terminals. Then I will discuss system changes in BTM and UTS we found necessary to support graphic terminals. Third, I will discuss graphic software, and finally I will mention our applications of graphics in teaching physics.

Review of Inexpensive Graphic Terminals

The last few years have shown a dramatic decline in the cost of graphics, and because of competing technologies this decline is likely to continue. A few years back it was difficult to do any type of graphics short of the 2250 type, costing well over $100,000, but the terminals I will describe have basic costs under $10,000. They do not provide all the facilities of a fancy dedicated graphics environment, but they offer a very usable set, much superior for many applications to alphanumeric abilities alone.

Perhaps the most widely used technology for graphic terminals is based on the Tektronix 611 storage display unit. Since the storage is in the phospher of the tube, no expensive core or

other memory is needed for refreshing the picture. The disadvantages are that selective erasing is not possible, and erasing takes one-half second. So we cannot show dynamically changing material. Both of the terminals we have used in our project, the ARDS 100A and the Tektronix 4002, are of this type. Several others are available, including the Computek and the Conographic. A spectacular price break recently occurred with the Tektronix announcement of the 4010, which sells in quantities of one for $3,950.

The Princeton Electronics Terminal is a second type of terminal, based on a selectively erasable memory tube. Its one-inch storage tube couples to a TV screen by means of a scan converter. The tube can also carry TV material. This terminal seems very promising, but I don't know how reliable it is.

A third inexpensive graphic terminal is based on the plasma display panel, as developed by Project Plato (University of Illinois) and Libby-Owens-Ford. So far no Plato terminals are available for general purchase; only one exists at the present moment within the project. But large numbers are said to be forthcoming soon. The source of light is a plasma discharge at individual point. The electronics of the display provide an inherent internal memory, and a point can be turned off as well as on. The terminals currently being produced require a 20-bit byte for input, and so would not be suitable for general purpose computers.

A fourth inexpensive
and perhaps internal
mini-computer with m
must be done from lo
be displayed is depe
how much terminal pr

Another strategy is
nals, usually TV set
is large, but when t
the cost is reasonab

Given these competin
position. This comp
graphics in the next
above, is already co
alphanumeric CRT ter

Graphics in BTM and
Although the termina
terminals under eith
ciently be used for
problem is that both
orientation, assumi
the terminals must
eight bits, sent ou
specify a point on

shing the picture. The disad-
sing is not possible, and erasing
nnot show dynamically changing
we have used in our project,
4002, are of this type. Several
the Computek and the Conographic.
tly occurred with the Tektronix
sells in quantities of one for

nal is a second type of terminal,
memory tube. Its one-inch storage
means of a scan converter. The
. This terminal seems very pro-
liable it is.

minal is based on the plasma dis-
oject Plato (University of Illinois)
no Plato terminals are available
exists at the present moment within
s are said to be forthcoming soon.
a discharge at individual point.
provide an inherent internal mem-
off as well as on. The terminals
re a 20-bit byte for input, and so
ral purpose computers.

A fourth inexpensive graphic terminal uses local refresh memory, and perhaps internal processing capabilities; the Imlac has a mini-computer with memory as part of the terminal. As refreshing must be done from local memory, the amount of material that can be displayed is dependent upon how much memory is present and how much terminal programming is used.

Another strategy is to use a disk to refresh a number of terminals, usually TV sets with keyboards. The cost of one terminal is large, but when the disk can be prorated to many terminals, the cost is reasonable.

Given these competing technologies, the user is in a fortunate position. This competition promises us better and cheaper graphics in the next few years. The Tektronix 4010, mentioned above, is already cost-competitive with the better hardcopy and alphanumeric CRT terminals.

Graphics in BTM and UTS

Although the terminals just described can be used as alphanumeric terminals under either BTM or UTS on the Sigma 7, none can efficiently be used for graphics without system changes. The basic problem is that both timesharing systems are alphanumeric in orientation, assuming that they are sending characters out, while the terminals must have carefully constructed combinations of eight bits, sent out just as is, in order to draw lines. To specify a point on the screen, four bites of information are

usually sent. These are structured in the software to give a 10-bit X and a 10-bit Y, plus auxiliary information such as whether the beam is on or off, whether the line is dashed, etc. (Generally several graphic modes are possible.) Hence the program must be able to send out any 8-bit code.

The Sigma timesharing systems place several obstacles in the way of getting such graphic output. Internal codes are EBCDIC while terminal codes are ASCII. Hence the monitor, thinking it is dealing with characters, "translates" from one code to the other. Furthermore the monitor may also check to see whether the character is a printable character, and refuse to send it out if it is not, or it may supply unwanted carriage returns. These "features" are lethal to graphics! We can attempt to untangle within our program the "translation" that the communication service has supplied, but this is clearly not efficient. We need simply the ability to send out any 8-bit code whatsoever, without translation, and nothing else.

In BTM terminal output, character by character, is done with a CAL3. To do graphics, our system programmers added several CAL4s to the system. They work in ways similar to the CAL3--they send out the "character" in register 0--but they do no code translation or checking for the printable character. These changes are relatively simple.

ed in the software to give a
iliary information such as
ether the line is dashed, etc.
re possible.) Hence the pro-
8-bit code.

ce several obstacles in the way
Internal codes are EBCDIC while
the monitor, thinking it is
tes" from one code to the other.
check to see whether the char-
nd refuse to send it out if it
carriage returns. These "fea-
e can attempt to untangle within
t the communication service has
efficient. We need simply the
e whatsoever, without transla-

er by character, is done with a
programmers added several
in ways similar to the CAL3--
register 0--but they do no code
printable character. These

Our experiences with UTS are more recent, and still incomplete. UTS in spite of its name is also oriented toward character IO. Terminal IO is done in a way very similar to disk and tape IO, through CALls and the use of DCBs, in particular M:UC. Again, as with BTM, translation is routinely done. Considerably more flexibility exists in principle, because we can specify terminal type; at present, however, only a few types are implemented. Our strategy in graphics is to use the M:DEVICE procedure to set a special bit in the DCB indicating that no translation is to be done, and then to output in the usual way.

Unfortunately this alone is not sufficient. As you may know UTS allows the user to specify a platen size. The default size is 72 characters. At this point UTS supplies, gratuitously, a carriage return and line feed in the IO stream. Unfortunately this is death to some graphics--the Tektronix terminal interprets a carriage return as indicating that it should go back to the character mode! So in the middle of the curve, letters of random description appear. We can set the platen size as a user, and so turn it off, but pedagogically it is unfortunate, to say the least, to start a program giving instructions to the user that may be meaningless to him. With somewhat more effort it can perhaps be set by a CAL, but only with some additions. If paging is the default option for your terminals, this will obviously also present problems, but at UCI we turn paging off for normal use. Again, this should be allowable in a CAL.

Although the above changes provide the possibility of UTS graphics, there does seem to be a way more consistent with the general UTS philosophy. A CAL allows you to specify the terminal type, so it would be natural to allow as a terminal type "graphics," meaning no translation or additional characters whatsoever. However this is somewhat a matter of taste, and UTS users may have other ideas as to what would be most natural.

In any case these system changes make it possible to do graphic output in either BTM or UTS. You will note that we have not discussed the question of graphic input, which seems to be a tougher problem. Although one of our graphic terminals has an inexpensive tablet associated with it, and both have joy sticks, we have not yet been able to use either of these devices. We will have to face some new problems besides just the question of avoiding the translation tables. We are worried, for example, that high speed graphic input, say as the student draws a curve, may swamp the input buffers.

Dialog Software for Graphics

Before reviewing basic graphic software, I will comment briefly about our software strategy in developing teaching materials, because graphic software is an extension of our already existing package. This package was described at a previous XDS Users Group meeting, and is fully documented. Those interested in further details should write or talk to me.

e possibility of UTS graphics,
istent with the general
specify the terminal type,
erminal type "graphics,"
characters whatsoever. How-
te, and UTS users may have
natural.

e it possible to do graphic
il note that we have not
ut, which seems to be a
graphic terminals has an
t, and both have joy sticks,
her of these devices. We
besides just the question
We are worried, for example,
s the student draws a curve,

vare, I will comment briefly
loping teaching materials,
nsion of our already existing
d at a previous XDS Users
ted. Those interested in
k to me.

Our teaching programs are METASYMBOL programs, sometimes with imbedded FORTRAN subroutines. The METASYMBOL programs are mostly procedure calls, each procedure call carrying out one of the functions necessary in a teaching program. These procedure calls were developed not abstractly but to meet the changing teaching needs reflected in the programs our teachers have wanted. At present we have about 150 procedures in our system. These link to a set of METASYMBOL library routines. To give you some idea of the flavor of the thing, here is a segment of a teaching dialog:

```
!EDIT STA11$
EDIT HERE
*TS325-350
          SKIP      1
          ENTRY
          WRITE     'WHAT DIFFERENTIAL EQUATION MUST BE SATISFIED BY'
          WRITE     'THE AMPLITUDE   A(X)'
          RESET     (LOOP,PROB)
N3        INPUT
          NOBLANK
          DELETEALL '(',')'
          BUMP      LOOP
          IF        'WAVE',S61
          IF        ('SHO','HARM','OSCIL'),BOR1
          IFNOT     '=',03
          IF        ('=0','0='),ZPOZ
          BUMP      MINUS
ZPOZ      IF        ('D12/DX12A','D12A/DX12','D12AX/DX12'),N4
          IF        'A''''',N4
          SAMEID    'EQ FOR A'
          BUMP      PROB
          SWITCH    PROB,(S67,S67,N5,S64)
S61       WRITE     'THE FULL EXPRESSION'
          WRITE     '      A (X)   EXP(IWT)'
          WRITE     'SATISFIES THE WAVE EQUATION. WRITE THE'
          WRITE     'EQUATION A SATISFIES.',N3
BOR1      WRITE     'YES, THE EQUATION IS SIMILAR TO THAT FOR THE'
          WRITE     'OSCILLATOR. ENTER IT!!',N3
03        TO        S62,(LOOP,2)
S63       WRITE     'I CAN NOT RECOGNIZE YOUR RESPONSE'
*
```

We do a good bit of internal file writing, for record keeping purposes, as the students run programs. We keep detailed files on what dialogs are used, what system errors occur, and what responses we could not analyze. We do not send error messages to the students. Most of our dialogs are far too long to fit into core in one piece, either BTM or UTS, so we support overlays. We allow the student, at the author's discretion, to leave a program before finishing and come back at the same point; a file stores the current status information, which can be extensive.

Graphic procedures were designed to augment our already existing teaching procedures. As in other developmental work, we first had graphic teaching programs running on a primitive basis, using manufacturer supplied FORTRAN routines that we adapted to the Sigma. Based on our experience, we then designed the procedures I will describe. You can find a full description of the graphic procedures in the Appendix. Here I will only try to give you a sample of what is available.

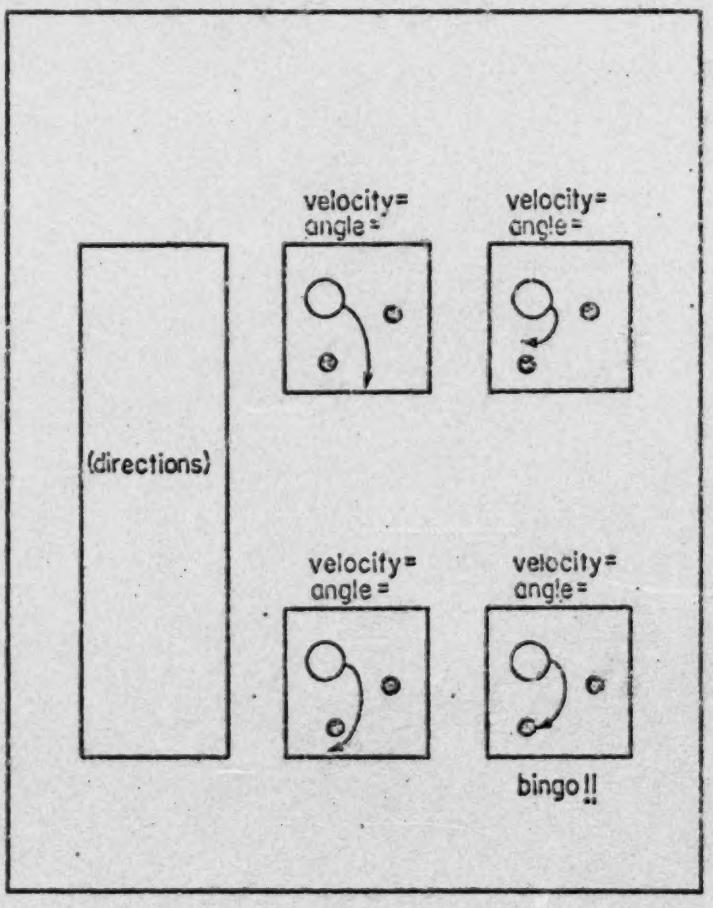
I will describe a hypothetical program, one that we have discussed but not developed. Suppose that we want to allow students to fire a lunar lander from the earth to the moon. Students are allowed to pick the initial velocity and the angle. The moon is rotating; we take its motion as a circle. We are seeing everything in the earth's frame of reference. We want them to launch their lunar landing module several times, perhaps profiting by previous misses of the moon. Hence we present a series of

writing, for record keeping
rams. We keep detailed files
tem errors occur, and what
do not send error messages
ogs are far too long to fit
or UTS, so we support overlays.
r's discretion, to leave a program
the same point; a file stores
ich can be extensive.

o augment our already existing
developmental work, we first
ning on a primitive basis, using
tines that we adapted to the
we then designed the procedures
full description of the graphic
I will only try to give you a

ogram, one that we have dis-
e that we want to allow students
arth to the moon. Students are
ity and the angle. The moon
s a circle. We are seeing
f reference. We want them to
several times, perhaps profiting
Hence we present a series of

pictures where we see the motion of his lunar landing device. Let's suppose that we allow as many as four shots. So four pictures will appear on the screen at one time. We also need auxiliary information, prompting the next velocity and direction, and indicating success or lack of success in the previous shot. So we need to divide the screen into four areas, or windows, and leave some room for writing messages. So the screen may eventually look like this:



The first graphic dialog command we issue is a DEVICE command:

DEVICE (ARDS,TEK)

The graphic coding for different terminals is different, so the program must know which terminal the student is using. Although at the moment we only support ARDS and Tektronix, a relatively small change in the programs would support other graphic terminals and add them as options. Thus the same program works on several different terminals.

Our next graphic command erases the screen, so that we can draw our orbits. It is obvious:

ERASE HOME

The option HOME indicates that the cursor is to be moved to the top of the screen, so now alphanumerical information will appear there.

Some description and instructions may now be written to the student; no graphics are involved, so we proceed.

Next we might ask the student for launching information. It is helpful to have this appear above the associated picture. To move the beam to a particular point on the screen we use a SETPOINT command:

issue is a DEVICE command:

minals is different, so the
student is using. Although
nd Tektronix, a relatively
upport other graphic termi-
the same program works on

screen, so that we can draw

cursor is to be moved to the
rical information will appear

ay now be written to the stu-
we proceed.

aunching information. It
e the associated picture.
int on the screen we use a


```
SETPOINT      (FS'2',FS'8')
```

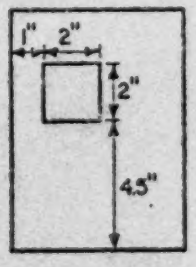
This refers to inches from the lower left corner of the screen. The different screens we use have different orientations, and so it may be necessary to specify this separately for each of the terminals, as in the following more complicated variant:

```
SETPOINT      (ARDS,(FS'2',FS'8')),(TEK,(FS'1.5',FS'5.5')
```

In a program, you would use only one of these commands. After the beam is placed, nongraphic commands query the student and store his input in appropriate locations.

Now we want to draw the orbit. We want the curve to fill not the whole screen but only a box in the upper left-hand corner, underneath the writing just done. We declare this area as a window for graphic output. We specify the window by giving distances in inches from the lower left for two corners, and again this specification can be different for different terminals.

```
WINDOW      (FS'1',FS'4.5'),(FS'3',FS'6.5'),BOX
```



The option BOX indicates that a box is to be drawn about the window.

Distances are necessary in specifying basic information for SETPOINT and WINDOW, but in plotting the curves we would like to use numbers in the form they occur within our computation. Let us suppose for the purposes of the computation we make the moon's 200,000 miles our distance "unit." We might choose differently. With this choice we might want to make a window roughly three units on each side, going from - 1½ to + 1½, both horizontally and vertically. Note that we have picked a square window, so we will not distort the picture. We specify the scale by the following procedure:

```
SCALE     (FS'-1.5',FS'1.5'),(FS'-1.5',FS'1.5')
```

The SCALE applies to whatever window has just been set; when we specify a new window, we also must remember to reset the scale.

So far we have no data to plot, even though we have queried the student about velocity and angle. The computation will be done by calling the FORTRAN subroutine. It is a typical numerical solution of a differential equation, with three gravitational forces: the sun's, which is essentially a constant, the earth's, and the moon's. A good first approximation is to assume the moon is moving on a circle. Remember that we are going to view things from the earth's frame of reference. On return from the

box is to be drawn about the

fying basic information for SETPOINT.
curves we would like to use num-
in our computation. Let us sup-
putation we make the moon's 200,000
night choose differently. With
e a window roughly three units
+ 1½, both horizontally and ver-
ked a square window, so we will
ecify the scale by the following

'),(FS'-1.5',FS'1.5')

indow has just been set; when
o must remember to reset the scale.

even though we have queried the
e. The computation will be done
ine. It is a typical numerical
tion, with three gravitational
ssentially a constant, the earth's,
approximation is to assume the
emember that we are going to view
of reference. On return from the

subroutine we have in arrays (say X and Y) the N points which represent the path of the lunar lander. We probably also would want to return with the position of the moon at the time the lunar landing crosses the moon's orbit, if it does; I assume that we do not plot the moon's circular path but that we simply show the initial position at the moment of launch and the final position at the orbit intersection.

So we want to draw three bodies before we start plotting the path: the earth and two moon positions. The first two will always be at the same location, (0,0) for the earth, and (1,0) for the moon. A small circle might be interesting for the three, but I won't go through the details.

Now we finally plot the path of the lunar landing. The following is all it takes:

```
CURVE    (X,Y,N)
```

It's not quite as simple as it looks; some experimentation may be needed within the FORTRAN program to get a reasonable curve. Remember that on our graphic terminal we draw a series of straight line segments. (Some terminals draw curved lines, but not those we have used.) Hence we want enough line segments so that they are sufficiently small and the curve looks reasonable, not choppy. On the other hand, if we put in too many segments, drawing the curve will take an enormous amount of time even at the 1200 baud

rate we use. Remember that four characters are generally sent for each new point on the curve, although some terminals allow short vectors that can be specified with less information.

This should be enough to give you an idea of the graphic procedures. Consult the Appendix for further details, as I have not covered all the Procedures. For example, we often want to draw labeled axes, so we do have a procedure for that. Also we want not just two-dimensional curves but we want projections in three dimensions, along with their axes. So the curve and axis routines allow specification of three variables, as well as two. The software takes care of the projections.

Graphics in Teaching Physics

Our last example, to show how to use the graphic procedures, also begins to display the power of graphics in physics teaching. All computer uses in education are at an early stage at present, and we have relatively less experience with graphics than with alphanumeric uses. Relatively little student experience exists with graphics; however the potentialities seem tremendous.

I will describe a graphics physics teaching program in classical mechanics, developed primarily by Richard Ballard of the Physics Computer Development Project. This program provides, with the motion of one particle systems, a range of experience for the student far beyond what he can acquire in his everyday life or in a well equipped physics laboratory. One of the difficulties

in teaching
to play with
initial cond

We can creat
even with th
modern devic
us to do a h
students any
than configu
and phase sp
particularly

When the stu
of terminal
and then he
directions,
ing nature.
elaborate t
the student
as a progra
what option
or doesn't
this progra
of view, fo

racters are generally sent
hough some terminals allow
with less information.

idea of the graphic proce-
ther details, as I have not
mple, we often want to draw
ure for that. Also we want
we want projections in three
So the curve and axis routines
les, as well as two. The
ons.

e the graphic procedures,
graphics in physics teaching.
at an early stage at present,
nce with graphics than with
le student experience exists
alities seem tremendous.

teaching program in classical
Richard Ballard of the Physics
s program provides, with the
range of experience for the
uire in his everyday life or
ory. One of the difficulties

in teaching about motion is that the student has little opportunity to play with the variables which affect how things move--forces, initial conditions, constants in the equations, masses, etc.

We can create only a few types of forces in a laboratory, and even with these it is difficult to eliminate friction, although modern devices such as the air track and the air table, allow us to do a better job than previously. We certainly cannot offer students any laboratory or practical experience with spaces other than configuration space, although such spaces as velocity space and phase space play a major role in our thinking about mechanics, particularly as we move beyond the most elementary considerations.

When the student first enters the program he is asked which type of terminal he is using, as indicated in our previous discussion, and then he is asked if he wants directions. If he requests directions, these turn out to be minimal, and mostly of an encouraging nature. The full facilities in this program are far too elaborate to explain initially. The directions given encourage the student to explore and try things, representing the program as a program of discovery. They tell him to ask questions or ask what options are available if he doesn't understand something or doesn't know what to do. Compared with many teaching programs, this program is extremely permissive from the student's point of view, for it does not "drive" him along a path.

Just what is drawn is dependent on what force law was selected. Whenever we come into a new force law, everything is set with some initial conditions, and the computer is always in a condition to plot something. In the harmonic oscillator program we initially plot position versus time for suitable initial conditions and the mass and spring constant both equal to one. In plotting at 1200 baud the curves come on at about the rate you would draw them by hand, so we have some feel of the process happening in time; curves do not appear instantaneously as with graphics at faster baud rates.

The student doesn't know the values of the various constants set, but he can ask questions. He can ask, "What is X?" or "What is X?" or "X = ?." A variable recognizer in the program can identify almost any physically meaningful variable, provided it can be preset and is not a dynamic consequence of the calculation.

He may start by exploring what happens if he uses different initial conditions than those originally specified. He may quickly plot dozens of curves with different initial conditions, until he is satisfied that he understands how varying initial conditions affect the motion. When the student wants to show several plots on the same screen (no erase between), he can use the term "overplot." He changes initial conditions by typing equations, just the way you would expect; thus he types "X = 2" to set the initial position to 2. He can also change the mass or spring constant or any physical quantity associated with the force he is exploring.

After directions he is as
different things to diffe
motions are available, or
way that seems reasonable
in physics by specifying
We don't ask for the forc
that is a primary method
many cases the student's
incomplete, and then we s
wants a central force, he
power law or a Yukawa for
a force the program check
and also provides him an
analytic expression conta

After the force is selec
If he is a true beginner
are available. If he ty
we record his requests i
improvement of the progr
randomly selected advice
of the facilities availa
times. Too much flounde
being requested. Many
to graph (or plot or loo
generated the graph.

After directions he is asked to select a motion. This can mean different things to different students. He can ask about what motions are available, or he can try to specify a system in any way that seems reasonable. Commonly we delineate different motions in physics by specifying the force, and that is what we want. We don't ask for the force, because we want the student to learn that is a primary method for identifying physical systems. In many cases the student's force specification turns out to be incomplete, and then we suggest the options available; if he wants a central force, he is queried as to whether he wants a power law or a Yukawa force. When he does successfully specify a force the program checks to see if it has understood him correctly, and also provides him an analytic expression for the force. This analytic expression contains the constants a student can change.

After the force is selected, the student is asked, "What now?" If he is a true beginner, he may have little idea of what facilities are available. If he types things we are unable to interpret, we record his requests in a disk file, for inspection later during improvement of the program, and we give him various pieces of randomly selected advice, designed to give him a better view of the facilities available. But we only let this happen a few times. Too much floundering leads to a graph being drawn without being requested. Many students suggest that they would like to graph (or plot or look at) the motion, so they will have already generated the graph.

…t force law was selected.
…, everything is set with
…ter is always in a condition
…cillator program we initially
…e initial conditions and
…al to one. In plotting
…out the rate you would draw
… the process happening in
…ously as with graphics at

…f the various constants set,
…, "What is X?" or "What is
…r in the program can identify
…able, provided it can be
…ce of the calculation.

…s if he uses different initial
…cified. He may quickly plot
…ial conditions, until he
… varying initial conditions
… wants to show several plots
…), he can use the term "overplot."
…ing equations, just the way
… 2" to set the initial position
…r spring constant or any physi-
…ce he is exploring.

It may then occur to the student to look at something besides positions. A natural possibility even for a beginning student is velocity versus time. He does this with "PLOT V,T" or "PLOT V VS T." A few brave students try to plot three variables at this stage, such as X versus V versus T, and are happy to discover that we do indeed cover this situation, plotting in perspective. However many students must be prodded to study motions in nonphysical spaces or with more than two variables.

Now let me mention auxiliary functions at the disposal of the student. It may turn out that a curve is all near the origin, not using most of the screen. In that case he or she may want to move closer to the picture, perhaps several times, to blow it up to reasonable size. Another scaling situation allowable is to change the scale separately on each of the variables. We allow that by using multiplication; if he wants to change the scale on the axis so that he shows 10 times the time, he says T*.1. Whenever the student makes any change the system responds with DONE to let him know that it has been recognized.

Several facilities are also provided to give ways of "reading" the three-dimensional plots. He can specify ROTATE, which move the axes around into the other positions, so that he is looking at the same curve from a different perspactive or he can ask for the negative values of one or all of the variables to be dashed, so that he can see when the curve goes through the coordinate planes. He can set up families of curves, where a series of curves

to look at something besides
even for a beginning student
this with "PLOT V,T" or "PLOT
to plot three variables at
ersus T, and are happy to discover
uation, plotting in perspective.
odded to study motions in nonphysical
iables.

ctions at the disposal of the
curve is all near the origin,
n that case he or she may want
erhaps several times, to blow
her scaling situation allowable is
on each of the variables. We allow
f he wants to change the scale
times the time, he says T*.1.
change the system responds with
s been recognized.

vided to give ways of "reading"
e can specify ROTATE, which move
positions, so that he is looking
rent perspective or he can ask
or all of the variables to be
n the curve goes through the coordinate
s of curves, where a series of curves

with some changes from one to the next are plotted over and over, without intermediate erasures. He can also alter the calculational details, of the numerical solution of the differential equation, altering the time step between the calculations; by looking at such things as kinetic energy versus potential energy he can check on the validity of the calculation.

So the dialog provides the student a rich mechanical environment, with great range of controllable possibilities. We can work for hours in any small subject area and only begin to see the possibilities. A number of visiting professionals have discovered interesting physical information, unknown to us before using the program. Thus when Gunter Schwarz from Florida State University was visiting, he took the two force center situation, used primarily as a model of a planet in a binary star system, and specified both forces as positive powers of the distance, the third or fourth. We were surprised to note that when we started the particle somewhere in the region between the two centers, the path of the particle was extremely like a Lissajou figure. An immediate terminal analysis, looking at force components, and a later analytic analysis revealed that to a good approximation each force component was linearly dependent on the corresponding position coordinate, within a wide range of choices of force law!

The binary star situation generally has been very good for getting interesting orbits, many of which are known to the professionals in that area, but hardly known to the general physics community.

We are now assembling a repertoire of such orbits to exhibit to visitors.

Graphics are absolutely essential for the type of teaching environment just described. Other graphics programs also exist, where we try to exploit the use of picture drawing in the teaching situation. I think you can see, even from this one example, why we are so excited about the future possibilities of graphics for teaching.

Acknowledgments

System support for graphics in both BTM and UTS were developed by Steve Slykhous and Joe Young, with the generous cooperation of David Sheldon. The graphics procedures were programmed by John Collins and Hal Deering with guidance from Estelle Warner and Alfred Bork.